

Unit - 2

Software Project management :=

* Effective software Project management is crucial to the success of any software project.

* In the past several projects have failed not for want of competent technical professionals or resources but because of use of faulty software Project management.

* The main goal of software Project management is to enable a group of software engineers to work efficiently towards successful completion of the project.

* For small software development projects, one of the software engineers assumes the responsibility of software Project management in addition to his normal responsibilities.

* The larger projects have full time Project management.

Responsibilities of a software project manager:

Software project manager mainly 2 types of responsibilities.

1) Job Responsibilities of a software project manager.

2) Skills Necessary for software project management.

1. Job Responsibilities of a software project manager:

* Software Project managers take the overall responsibility of steering a project to success.

* The job responsibilities of a project managers range from invisible activities like building up team morale to highly visible customer presentations.

* Most managers take responsibilities to project proposal writing, project cost estimation, scheduling, project staffing, software configuration management, risk management, interfacing with clients, managerial report writing and representations.

Skills necessary for software project management:-

=> A theoretical knowledge of different project management techniques is certainly necessary to become a successful project manager.

=> The effective software project management frequently calls for good qualitative judgment and decision making capabilities.

=> Having a good grasp of the latest software project management techniques such as cost estimation, risk management, configuration management, project managers need good communication skills.

=> Some skills such as tracking and controlling the progress of the project, customer instruction, managerial presentation.

28/10/2019

Project Planning :=

⇒ Feasible study.

⇒ Before developing sw product.

⇒ Estimate.

⇒ size estimate

⇒ duration.

⇒ cost.

⇒ scheduling.

⇒ staff.

* once project is found to be feasibility study, software project manager under take project planning.

* Project planning is completed even before development activity starts.

* Project planning consists of the following essential activities.

* estimating some basic attributes of the project.

1. duration (how long will take to complete the project).

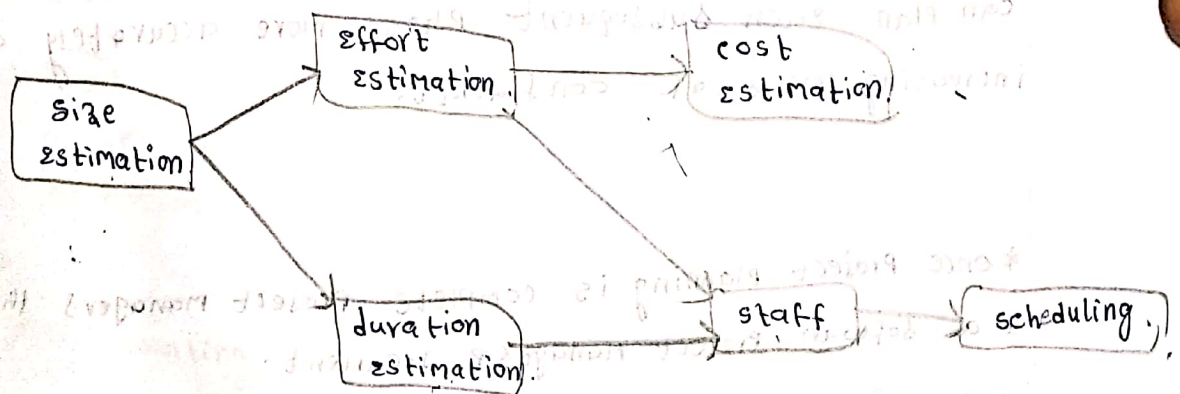
2. effort (how much effort would be required).

3. cost

4. scheduling.

5. staff.

Planning activities of a project:



* observe that size estimation is the first activity.

* it is also the most fundamental parameter based on which all other planning activities are carried out.

* other estimations such as effort, cost, duration, staff, scheduling are also important components of project planning.

* Project planning requires utmost care and attention since commitment to unrealistic time and resource estimation result in schedule slippage.

* schedule delays can cause customer dissatisfaction and adversely affect team morale. it can even cost project failure.

* especially for large projects it is very difficult to make a plan. a part of this difficulty is due to the fact that the project parameters, scope of the project, project staff etc. may change during the span of the project.

* in order to overcome this problem some times project managers undertake project planning in stages.

* This technique of staggered planning is known as sliding window planning.

* sliding window technique starting with an initial phase.

* The project is planned successively more accurately successive

* after the completion of every phase the project managers can plan each subsequent phase more accurately and with increasing levels of confidence.

* once project planning is complete project managers then plan in a software project management document.

1) Introduction : objectives.

major functions.

management & technical constraints.

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

Estimations: cost

size

Duration.

effort.

SPM Techno

(3) Resources: people.

software.

Hard ware.

(4) scheduling: work Breakdown structure, Gantt chart, PERT chart.

(5) Staff: team structure.

(6) Risk management: Risk Analysis, Risk Reduce, Risk identification, Risk estimation.

(7) Project control & tracking.

(8) Miscellaneous: Verification & Identification.

=> Quality assurance.

=> Installation.

=> Delivery.

=> Maintenance.

29/1/19. Project size estimation:

=> cost, effort, duration.

=> estimate size

=> LOC => (lines of code).

=> FP => (function point).

1) LOC:

only counting execution part not counting head files, comments blank spaces.

=> LOC is simplest among all metrics available to estimate project size.

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సర్వే

=> using this, The project size is estimate by counting
The number of source ~~text~~ instructions in the developed Program:

=> Determining the LOC count at the end of a project is very simple job. however accurate estimation of the LOC count at the beginning of a project is very difficult.

=> LOC gives a numerical value of problem size that can vary widely with individual coding style, different program's layout their code in different ways.

=> A good problem size measure should consider the overall complexity of the problem and the effort needed to solve it.

=> LOC measure correlates poorly with the quantity and efficiency of the code.

=> LOC is very difficult to accurately estimate. LOC in the final product from the problem specification. The LOC count can be accurately computed only after the code has been fully developed.

=> LOC is any line of program text, that is not comments or blank spaces. regardless of number of statements on the line.

Ex:

1. #include <stdio.h>

2. int main()

3. {

4. int a, b;

5. printf("%d", a+b);

6. return 0;

7. }

LOC = 7.

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

Function Point :-

* function point metric was proposed by albert.

* function point metric is that it can be used to easily estimate the size of a software product directly from the problem, specification.

* The conceptual idea underlying the function point metric is that the size of software product is directly dependent on the no. of different functions or features it supports,

* it is structured technique that decomposes systems down into smaller components so that they can be better understood and analyzed.

⇒ function point based estimates are based on the following five information domain values.

1. No. of inputs.

2. No. of outputs.

3. No. of inquiries.

4. No. of internal logical files.

5. No. of external interfaces.

⇒ function point estimation is performed in the following

calculate the unadjusted function point (UFP) :

it is calculated by simply counting the value of each with each information domain and multiplying.

complexity adjustment attribute (CAA):

$$CAA = 0.65 + 0.01 \times EF_i$$

EF_i = total complexity adjustment value.

$i=0$ -> No significant

$i=1$ incidental

$i=2$ moderate

$i=3$ average

$i=4$ significant

$i=5$ essential

$$FP = UFP \times CAA$$

information
domain.

simple

avg

com.

ill

3

4

olq.

4

5

inquiry

3

4

Inter

7

10

15

exter.

5

7

10

Project Effort Estimation:- Techniques:-

→ cost, Engineers, Time.

⇒ cost:- s/w, h/w, Employee salaries, Tools.

⇒ 3 techniques

1. Heuristic Estimation

1. Effort Estimation predicts how much time is required to complete a project, how much it cost, how many engineers are required for completing the project.

2. cost effort includes the cost of s/w, h/w and salaries of engineers incurred other cost including the training, travels, support tools etc.

3. Effort Estimation is imp for cost. benefit determination, Project scheduling and monitoring & control.

4. The most considerable effort is the requirement of skilled engineers. therefore it is generally estimated in terms of person months. It

there are 3 broad categories of estimation techniques

1. Heuristic Estimation technique.

2. Empirical estimation. "

3. Analytical. " "

1. COCOMO Heuristic Estimation Techniques:-

⇒ Constructive Cost Estimation Model.

⇒ Barry bahem (1981)

⇒ it developed for effort calculating effort, development time, persons.

⇒ COCOMO model applied 3 classes on software project.

1. organic - ^{Project size} 2-50 KLOC - ^{nature of project} simple - ^{Deadline of project} not tight schedule.

2. Semidetached mode - 50-300 KLOC - medium - medium

3. Embedded ^{సాంఘిక మరియు గిరిజన సంక్షేమ కార్యాల వారి ఉచిత సహకార}
⇒ above 300 KLOC - ^{complex} - within time

COCOMO model categorized into 3 types

1. Basic cocomo
2. intermediate cocomo
3. complete cocomo model.

or

(1) Basic COCOMO

It estimates software development time, effort, No. of persons.

Equations are

$$\Rightarrow E = a_b (KLOC)^{b_b}$$

$$\Rightarrow D = c_b (E)^{d_b}$$

$$\Rightarrow P = E/D$$

$$E = a_b (KLOC)^{b_b}$$

$$D = c_b (E)^{d_b}$$

$$P = E/D$$

where E = effort applied in person month.

D = Development time (months).

P = total number of persons.

a_b, c_b, d_b are coefficients.

Software Project	a_b	b_b	c_b	d_b
Organic.	2.4	1.05	2.5	0.38
Semidetached mode	3.0	1.12	2.5	0.35
Embedded mode.	3.6	1.20	2.5	0.32

1. Suppose that a project was estimated to be 400 KLOC. Calculating the effort, development time, persons estimation for organic mode?

Sol: $E = 2.4 (400)^{1.05} = 1295.31 \text{ PM}$

$$D = 2.5 (1295.31)^{0.38} = 38.07 \text{ M}$$

$$P = 1295.31 / 38.07$$

$$= 34.02 \text{ P.}$$

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

attest 2

in semidetached mode.

$$E = 3.0(400)^{1.12} = 2462.7 \text{ PM}$$

$$D = 2.5(2462.7)^{0.35} = 38.45 \text{ M}$$

$$P = 2462.7 / 38.45 \\ = 64.04 \text{ P}$$

③ in Embedded mode.

$$E = 3.6(400)^{1.20} = 4772.8 \text{ PM}$$

$$D = 2.5(4772.8)^{0.32} = 37.59 \text{ M}$$

$$P = 4772.8 / 37.59 \\ = 126.9 \text{ P}$$

2. Suppose that a project was estimated to be 30,000 LOC calculate Effort, development time, Persons, using semidetached Mode?

Sol:

Given data = 30,000

$$E = a_b (KLOC)^{b_b} \\ = 3.0(30)^{1.12} \\ = 135.36 \text{ PM}$$

$$D = c_b (E)^{d_b} \\ = 2.5(135.36)^{0.35} \\ = 13.9 \text{ M}$$

$$P = 135.36 / 13.9 \\ = 9.7 \text{ P}$$

(2) Intermediate COCOMO Model :-

⇒ it is extension of basic COCOMO model.

⇒ 15 ⇒ "cost drivers attributes". To compute the cost of SW.

⇒ ^{Divided} 4 types attributes.

1. Product attribute. → 3

2. Hardware. → 4

3. Personal. → 5

4. Project. → 3.

⇒ ~~15~~ "CDA" or

⇒ Product attribute :-

1. Software reliability. (RELY).

2. Size of Database (DATA).

3. Product complexity (CPLX).

⇒ Hardware attributes :-

1. Runtime Performance constraints (TIME).

2. Memory storage constraints (STORE)

3. Virtual storage constraints (VIRT).
machine. Volatility.

4. Require turn about time (TURN).

⇒ Personal attributes :-

1. Analyst capability (ACAP).

2. Applications experience (AEXP).

3. Programmer capability (PCAP)

4. Virtual machine Experience (VEXP)

5. Programming language Experience (LEXP).

⇒ Project attributes :-

1. Modern Programming practices (MODP).

2. Use of Software tools (TOOL).

3. Development schedule (SCHED).

* 15 attributes get a 6 point scale ranging from very low to extra high.

* Each cost drivers and ratings can be viewed as very low, low, Normal, High, very high, extra high.

* Intermediate model equations.

$$E = a_b (KLOC)^{b_b} \times EAF \quad (\text{Effort Adjustment factor}). PM$$

$$D = C_b (E)^{d_b} M$$

$$P = E/D \cdot P.$$

* And the effort of eDA is given following table. The product of all effort multipliers result in EAF.

Name	very low	low	normal	high	very high	Extra high.
1. RELY	0.75	0.88	1.00	1.15	1.40	-
2. DATA	-	0.94	1.00	1.08	1.16	-
3. CPLX	0.70	0.85	1.	1.15	1.30	1.65
4. TIME	-	-	1.00	1.11	1.30	1.66
5. STORE	-	-	1.00	1.06	1.21	1.56
6. VIRT	-	0.87	1.00	1.15	1.30	-
7. TURN.	-	0.87	1.00	1.07	1.15	-
8. ACAP	1.46	1.19	1.00	0.86	0.71	-
9. AESP	1.29	1.13	1.00	0.91	0.82	-
10. PCAP	1.42	1.17	1.00	0.86	0.70	-
11. VEXP	1.21	1.10	1.00	0.90	-	-
12. LEXP	1.14	1.07	1.00	0.95	-	-
13. MODP	1.24	1.10	1.00	0.91	0.82	-
14. TOOL	1.24	1.10	1.00	0.91	0.83	-
15. SCHED.	1.23	1.08	1.00	1.04	1.10	-

1. A project having 30,000 LOC which is an embedded s/w with critical area s/w reliability is high. estimate the devlop time, effort. No. of persons?

Sol:

$$30,000 \text{ LOC} \Rightarrow 30 \text{ KLOC.}$$

$$E = 3.6(30)^{1.20} \times 1.15$$

$$= 245.21 \text{ PM.}$$

$$D = C_b(E)^{0.32}$$

$$= 2.5(245.21)^{0.32}$$

$$= 14.54 \text{ M.}$$

$$P = E/D$$

$$= 245.21 / 14$$

$$\Rightarrow 16.86 \approx 17.$$

2. A Project having 40 KLoc which is an org organic s/w.

with critical area s/w Analyst capability is Normal estimate dt, E, P.

Sol:

$$40 \text{ KLoc.}$$

$$E = 2.4(40)^{1.05} \times 1.00$$

$$= 115.44 \text{ PM.}$$

$$D = C_b(E)^{0.35}$$

$$= 2.5(115.44)^{0.35}$$

$$= 13.19.$$

$$P = E/D$$

$$= 115.44 / 13.19 = 7.599 \approx 8.$$

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

Detailed COCOMO:-

⇒ A major short coming of both basic and intermediate COCOMO models is that consider a software product as a single homogeneous entity.

⇒ The most large project are made up several similar sub projects.

⇒ 4 phases used in Detailed COCOMO.

1. Requirement Planning & product Design. (RPD).
2. Detailed Design (DD).
3. coding & unit testing (CUT).
4. IP Integration testing. (IT).

⇒ Equations are:

$$E = a_b (KLOC)^{b_b} \times EAF.$$

$$D = c_b (E)^{d_b}.$$

$$P = E/D.$$

Names Effort & A.	very low	low	normal	high	very high	extra high
1. RPD	1.80	0.85	1.00	0.75	0.55	-
2. DD	1.35	0.85	1.00	0.90	0.75	-
3. CUT	1.35	0.85	1.00	0.90	0.75	-
4. IT	1.50	1.20	1.00	0.85	0.70	-

1. Consider a project having 50,000 LOC which is organic mode software area is RPD & time is high estimate E D P.

Sol:

50,000 LOC

⇒ 50 KLOC.

$$E = a_b (KLOC)^{b_b} \times EAF$$

$$= 2.4 (50)^{1.5} \times 0.75$$

$$= 109.44 \quad \text{సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా}$$

$$D = C_b(E)^{0.38}$$

$$= 2.5(109.44)^{0.38}$$

$$= 14.88 M.$$

$$P = E/D$$

$$= \frac{109.44}{14.88}$$

$$= 7.35 \approx 7. P$$

2. Consider a project having 30KLOC which is semidetached mode software area. Integration testing is low & all mode E D P 9.

sol,

$$E = 3.0 (30)^{1.12} \times 1.20$$

$$= 162.43 PM$$

$$D = 2.5(162.43)^{0.35}$$

$$= 14.84 M.M$$

$$P = E/D$$

$$= \frac{162.43}{14.84} = 10.94 \leq 11 P.$$

Empirical estimation techniques:-

=> Experts guess (Educate); of the project parameters.

(1) Expert judgement:

→

=> while using this technique prior experience with the development of similar product are helpful.

=> 2 popular estimation techniques are

1. Expert judgement technique
2. Delphi cost estimation

సాంఘిక మరియు గణిత సంక్షేమ శాఖల వారి ఉచిత సరఫరా

Expert judgement technique:

1. Expert makes an educated guess of the problem size after the problem thoroughly.
2. The expert estimates the cost of the different components (that is modules are subsystems of the system) ~~the~~ and then combines them to arrive at the overall estimate.
3. This technique is subject to human errors

2. మొదటి భాగం 1992 నాటిది. దీనిలో
మొదటి భాగం 1992 నాటిది. దీనిలో
మొదటి భాగం 1992 నాటిది. దీనిలో
మొదటి భాగం 1992 నాటిది. దీనిలో
మొదటి భాగం 1992 నాటిది. దీనిలో

Software Design

Characteristics of good S/W Design

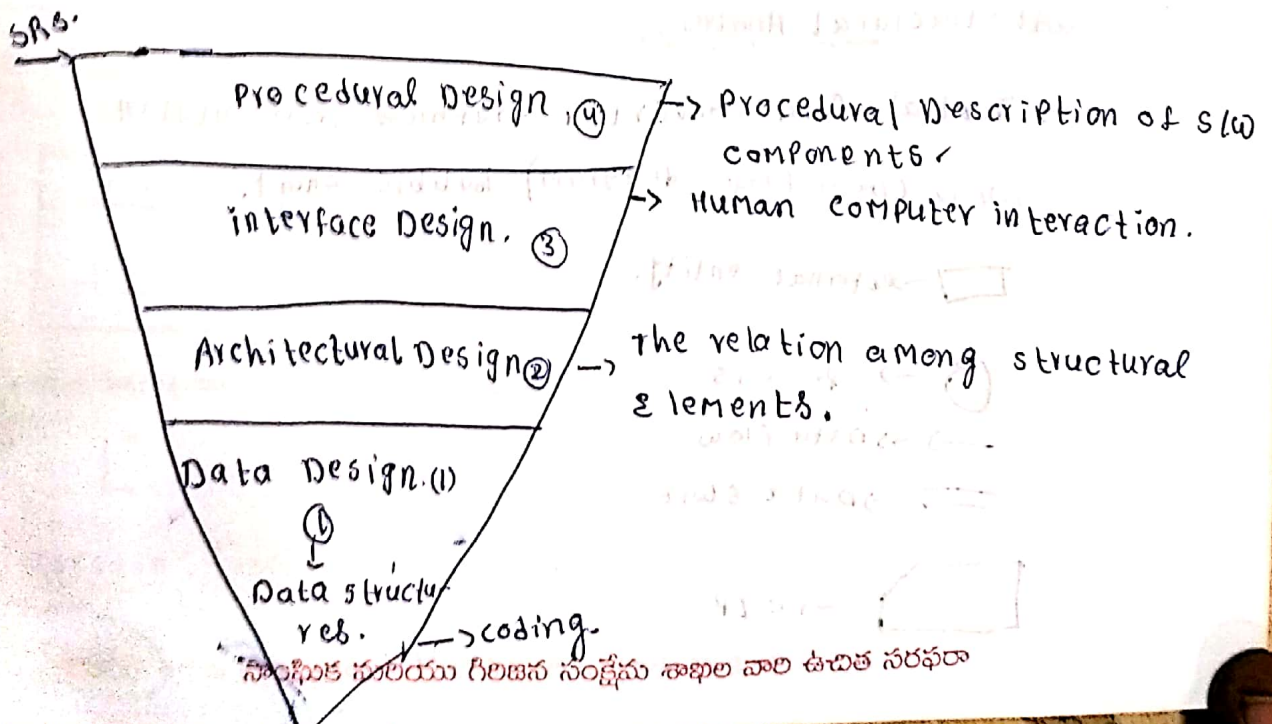
1. Maintainability
2. Reusability
3. Reliability
4. Portability
5. Efficiency
6. Correctness
7. Simplicity
8. Modularity
9. Verifiability
10. Completeness
11. Understandability

S/W Design Phases

3 Types steps

1. Decompose the system into modules.
2. Determine Relation b/w the modules.
3. Describe individual module functionalities.

Levels of S/W Design



Function and - object oriented Design:

Function oriented Design.	object oriented Design.
1. it mainly focus on functions.	1. it mainly focus on objects.
2. it mainly emphasize on what system does.	2. what the system consists of.
3. Methodologies are SDLC & incremental; methods structured Analysis.	3. incremental; methods.
4. High Risk.	4. Low Risk.
5. low Reusability.	5. High Reusability.
6. well defined Projects with stable user requirements.	6. well defined Projects with changing user requirements.
Ex: ATM.	Ex. TV.

Function oriented Design: 2.

1. Top down decomposition.
2. centralized system design.

SA: structural Analysis:

=> Textual form converted graphical representation

=> DFD (Data flow diagram) bubble chart.

□ - external entity.

○ - process.

→ - data flow.

⇒ - data store.

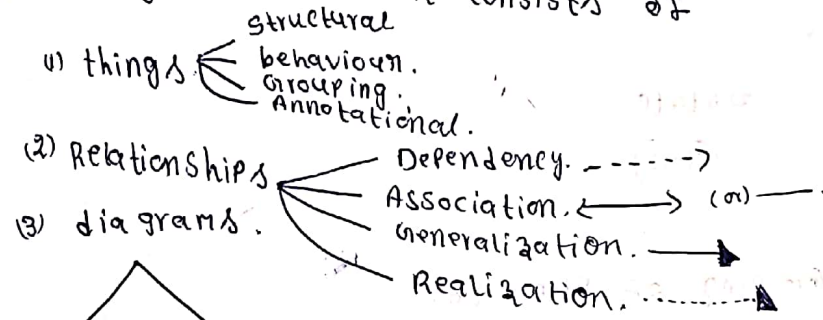
▭ - OLP

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

UML - unified modelling language.

1. building blocks of UML.
2. Rules to connect building blocks of UML.
3. mechanism of UML.

* building blocks of UML consists of



Structural

1. class.
2. object.
3. component.
4. Deployment.

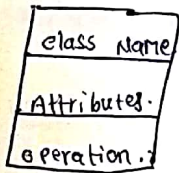
Behavioural.

1. usecase.
2. statechart.
3. Activity.
4. sequence.
5. interaction.

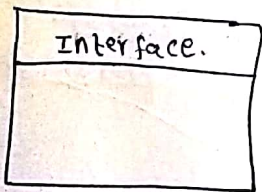
Things:

(1) structural things :-

1. class : blue print of object.



2. Interface : responsibilities of a class.



3. Collaboration : interaction between the elements.

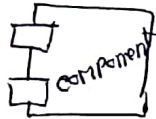


4. usecase : functionalities of a system.

use case

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సర్ఫరా

5. component :



physical (or) software.

6. Node : server representations.

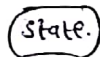


Behavioral things :-

1. interaction.



2 state.



Grouping things: Grouping of elements.

it defines package.



Annotation :

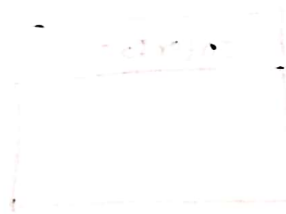
it defines note. write comments.



=> Relationships:

1) Dependency :-

represents - - - - ->



Design Patterns:-

reusable & repeatable solution to the frequently occurring problems in design phase.

⇒ introduced by GOF (Gang of four).

1. Erich Gamma.
2. Richards Helm
3. Ralph Johnson.
4. John Vellisides.

⇒ we have 23 Design Patterns in S.G.

1. Creational Pattern. (5) ⇒ object creation.
2. Structural Pattern. (7) ⇒ composition.
3. Behavioural Pattern. (11) ⇒ Interface.

Every design having 4 elements.

- ⇒ Pattern name.
- ⇒ Problem.
- ⇒ solution.
- ⇒ consequence.

1. Creational Patterns: 5.

1. Prototype:- A full rules and regulations of instance object to be copied.
2. Build:- Separate the objects from construction from its representation.
3. Singleton:- A class which only ^{have} single object can exist. Ex: restaurant.
4. factory:- A specific instance of the factory method.
5. factory methods:-
collection of the objects.

2. Structural Patterns:- 7.

1. Proxy:- one object represents another object. Ex: ATM.
2. Decorate:- add responsibilities to another object dynamically.
3. Flyweight:- An object used for efficiently sharing at a time.

Ex: Telephone

సాంఘిక మరియు గౌరవన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

4. facade: A single class represents the entire subsystem.

Ex: Event manager.

5. Bridge: Separate an abstraction from its implementation.

6. composite: it combines two or more similar objects into single objects.

7. Adapter: match the interfaces of different classes.

8. Behavioural Patterns := (11).

1. chain responsibilities := a way of passing request b/w a chain of objects.
Ex: applying a loan.

2. iteration := sequentially access the elements from the collection.
Ex: TV Remote.

3. state := modify objects behaviour when its state changes.
Ex: Fan and its controller.

4. strategy := encapsulate an algorithm inside a class.
Ex: Sorting a number.

5. observer := A way of notifying change for a number of classes.
Ex: online bidding, cricket points.

6. Visitor := Define a new operation to a class without changing.
Ex: Taxi.

7. template method := Defines exact steps of an algorithm.
Ex: house plan.

8. command := encapsulates a command request to an object.
Ex: server in hotel.

9. mediator := communication b/w two or more classes.
Ex: Airline control system.

10. Memento := capture & restore the objects internal state.
Ex: Games.

11. interpreter := implements specified language is called as interpreter-preter.

Uses of DP:

1. Make it easy to reuse successful Design and architecture.
2. Make it easy for New developers. to big design software.
3. Helps in documenting and maintain software.

అధ్యయనం

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

అధ్యయనం అంటే ఏమిట?

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

: unit-4:

1) coding & ^{unit} testing.

Syllabus:

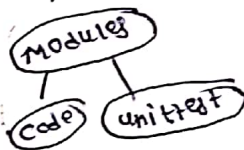
1. implementation.
2. coding standards & Guide lines.
3. Code review.
4. Verification & Validation.
5. unit testing.

1. implementation:- (coding):

=> coding is undertake ^{successfully} after design phase.

=> coding I/P Design phase. O/P I/P system testing.

=> Design phase



```
graph TD; A[Design phase] --> B(modules); A --> C(code); A --> D(unittest);
```

unit-testing: Testing every individual module with^{out} effecting the other module.

=> after completion of coding & unittest → goto Integration & system

=> Design

=> Def:- Transfer Design document into code in any high level language Then Perform unit Testing.

=> coding standards

=> coding guideline

=> Code review.

=> Find the errors before Testing

→ find the errors before testing

Coding standards & guidelines:

⇒ every individual organizations having diff Rules and Regulations.

⇒ Coding standards:

1. Limit usage of Global.
2. Standard header Modules.
3. Naming Global Variables.
4. local Variables.
5. constant identifier.
6. convention error returning values.

Coding Guidelines:

1. code should not write too clever (or) too difficult.
2. avoid side effects.
3. can't use identifiers in multipurpose.
4. well defined document.
5. function code should not exceed 10 lines of source code.
6. avoid goto statements.

Code Review:

in code review we can detect the error before testing.

1. code walkthrough.
 2. * successfully of unit testing & remove syntax error.
- * cost effective solution.

* Types 1) code walkthrough.

2) code inspection.

code walk through:

⇒ After completion of unit testing & syntax error code walk through was started.

⇒ Development team → few days before code

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

unit-3

* cohesion & coupling

* food & food

* Design patterns.

unit-4

* implementation.

* standards & guidelines

* code review.

* unit testing,

→ code walk through meeting → Present coder of module.



Rectify logical errors, algorithmic error.

Guidelines of code walk through:-

* The team has min 3 members & Max 7 members

* errors find.

* manager not be attended.

code inspection:-

→ checking common & standard errors

1. uninitialized variable

2. array index out of the bound.

3. mismatch Actual parameters & formal.

4. Allocation & deallocation.

5. terminating loops, jump loops.

Clean room testing:

* CW, CI, formal verification

* IBM company has developed the clean room testing.

* it prepare document.

* Disadvantage:- Time consuming

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సర్వీసు

Cohesion & coupling:

=> cohesion & coupling are necessary in making any s/w reliable & extensible.

Cohesion:-

it is nothing but the ^{measure of} functional strength of a module.

=> in general it measures the relationship strength between pieces of functionality within a given module in s/w programming.

=> The measurement leads to be

high cohesion.

1. in a good module, various parts having high cohesion is preferable due to its

1. reliability.
2. reusability.
3. robustness &
4. understandability.

low cohesion.

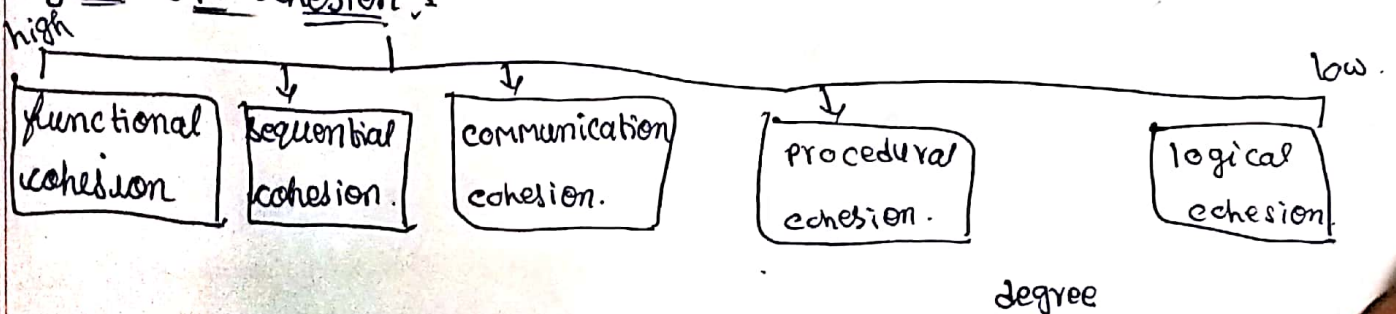
associated with undesirable traits including difficult is

1. maintaining
2. reusing &
3. understanding.

=> High cohesion Advantages:

- * High cohesion leads to increase module reusability.
- * System maintainability will increase.
- * Module complexity also reduces.

Types of cohesion:-



1. functional cohesion:-

in which the parts of modules are grouped b/w they all contribute to the modules single well defined task.

2. sequential cohesion:-

→ where the parts of modules grouped due to the output from one part is the input to another.
Ex: withdraw money.

3. communicational: The parts of the module are grouped because they operate on same data.

4. procedural: The parts of the modules are grouped together b/c a certain sequence of execution is followed by them.
Ex: algorithms.

5. logical: All the elements in the module perform similar operations.
* logical cohesion exists when logically related elements of module are placed together.

coupling:

* it indicates closely two modules interact of how interdependent they are.

⇒ The degree of coupling b/w two modules depends on their interface complexity.

⇒ if the system has low coupling, it is a sign of well structured computer system and a great design.

⇒ if low coupling with high cohesion, it supports the mission of high reliability.



modular Design:

- => message C, C++.
- => Data
- => Stamp => structures in C
- => control => flag.
- => external
- => content.
- => common. => static global.

cohesion.

1. functional.
2. sequential.
3. Communication
4. logical.
5. procedural
6. temporal.
7. coincidental.

Guidelines and standards.

Standards:

do not write code which is too difficult (or)

Codes are having some standards and guidelines.

Guid lines:

1. code should not write too clever (or) difficult
2. avoid side effects.
3. Do not identify in multipurpose.
4. we prepare own document
5. Code do not exceed 10 lines of source code.
6. avoid goto statements.

standards:

1. limit usage of global.
2. Naming the global variable.
3. local variable.
4. constant identifiers.
5. convention of error returning value
6. standard header limit.

code review: detect the error before testing & successfully of unit testing & remove syntax errors.

1. code walkthrough.
2. code inspection.

⇒ after successfully completion of unit testing & syntax error clear

1. The code is given to development team before few days.
2. Code walkthrough meeting present code.

↳ Rectify the logical error, algorithmic error.

1. uninitialized variables.

2. array index out of the bound.

3. mismatch variable and actual parameter.

4. Allocation & deallocation.

5. terminating loops and jump loops.

Verification & Validation:

Verification: check executable work product.

Validation: = executable software product.

→ (V & V) software production specification customer satisfied

→ (V & V) quality.

→ RAS → design → coding inspection.

→ Barry Bohem said that

Verification: Are we building right product.

Validation: Are we building product right.

* IEEE def for Verification:

inspection, coding, auditing, testing, review.

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

unit-5

*

Integration & system testing

* Integration testing.

* system testing.

* Black Box Testing.

* white Box testing.

* Debugging techniques.

Integration testing:

i/p :- coding & unit testing.

=> it is process combine all integrated Modules (units).

=> find errors.

4 types => (1) Big Bang i/t.

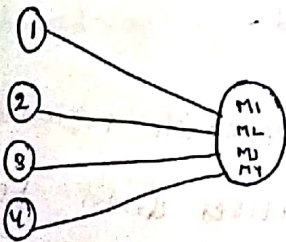
(2) Top down i/t.

(3) Bottom up i/t.

(4) Sandwich i/t (mixed).

(1) Big-Bang i/t:

* if we have 4 modules to combine single module.



=> large programs. Not used.

అవగాహన

సాంఘిక మరియు గణిత సంక్షేమ శాఖల వారి ఉచిత సరఫరా

Top down it.

Bottom top

* stub - dummy module

* combining all the modules once and verifying the functionality after completion of individual module testing.

* in this approach, all the modules making up a system are integrated in a single step.

* This technique can meaningfully be used only for very small systems.

* The main problem with this approach is that once a failure has been detected during integration testing, it is very difficult to localise the error as the error may potentially lie in any of the modules.

* Debugging errors reported during big-bang testing are very expensive.

* Big bang approach almost never used for large programs

Top-down integration Testing:

=> Top-down integration testing starts with root module in the structure chart and one or two subordinate modules of the root module.

=> After the top level skeleton has been tested, the modules that are at the immediately lower layer of the selection are combined with it and tested.

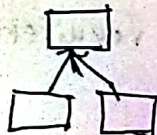
=> Top-down integration testing approach requires the use of program stubs by the routines under test.

=> A disadvantage of topdown integration testing approach is that in the absence of lower level routines, it becomes difficult to exercise the top level routines in the desired manner once the lower level routines usually perform input/output (I/O) operations.



Bottom-up integration testing:

- => large software products are often made up of several subsystems.
- => A subsystem might consist of many modules which communicate among each other through well defined interfaces.
- => in bottomup integration testing, first the modules for the each subsystem are integrated thus the subsystems can be integrated separately and independently.
- => in a pure bottom up testing no stubs are required and only drivers are required.
- => large software systems normally require several levels of subsystem testing. lower level subsystems are successively combined to form higher level systems.
- => The advantage of Bottom up Testing is that several disjoint subsystems can be tested simultaneously.
- => The disadvantage of bottom of Testing is the complexity that occurs when the system is made up of large number of small subsystems that ~~are~~ are at the same level.



సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

4) mixed (sandwiched) i/t:

* The mixed integration testing follows a combination of topdown & bottomup testing approaches.

* in topdown approach, testing can start only after the top level modules have been coded and unit tested.

* in bottomup approach, testing can start only after the bottom level modules are ready.

* The mixed approach overcomes this short coming of the topdown and bottom up approaches.

* in the mixed testing approach, testing can start as and when modules become available after unit testing.

* in this approach both stubs and driver are required to be designed.

System testing:

=> After completion of the integration testing.

1) Alpha: Development within the organization.

2) Beta: friendly customers.

3) acceptance: customers.

* before system testing we do smoke testing.

=> one main functionality.

* System testing 2 types.

1. functionality:

2. Performance test: Non-functional requirements.

1. stress testing $\Rightarrow$ within time, capacity.
2. Volume testing $\Rightarrow$ size, area. (Data structures). ex. databases.
3. compatibility testing $\Rightarrow$ s/w & h/w supports testing (interfaces).
4. configuration $\Rightarrow$ h/w & s/w requirements configuration.
5. Recovery $\Rightarrow$ Data loss, faults recovery.
6. Regression $\Rightarrow$ maintenance errors & functionalities. (add).
7. maintenance $\Rightarrow$ a dividing functionalities (procedures).
8. Documentation $\Rightarrow$ manuals (Technical, code).
9. usability $\Rightarrow$ user understanding (screens, msg).
10. security $\Rightarrow$ saving, authentication (Theft Data).

$\Rightarrow$ Error Seeding $\Rightarrow$

$\Rightarrow$ Sometimes customers specify the maximum number of residual errors that can be present in the delivered software.

$\Rightarrow$ These requirements are often expressed in terms of maximum number of allowable errors per line of source code.

$\Rightarrow$ The error seeding technique can be used to estimate the number of residual errors in software.

$\Rightarrow$ it involves seeding the code with some known errors.

$\Rightarrow$ The number of these seeded errors that are detected in the course of standard testing is determined.

These values in conjunction with the number of unseeded errors detected during testing can be used to predict,

$\Rightarrow$ the number of error remaining in the product.

$\Rightarrow$ The effectiveness of the testing strategy.

$$\frac{n}{N} = \frac{S}{S}$$

$\Rightarrow$ let 'N' be the total number of defects in the system.

$\Rightarrow$ let 'n' of these defects be found testing.

$$N = \frac{S \times n}{s}$$

$\Rightarrow$ let 'S' be the total number of seeded defects.

$\Rightarrow$ let 's' of these defects be found during testing.

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సేవ

⇒ Defects still remaining in the Program after testing can be given by

$$N-n = n \times \frac{(s-1)}{s}$$

⇒ error seeding works satisfactorily only if the seeded errors match closely with the kind of defects that actually test.

Black-Box testing:

⇒ to examine input values in a slw.

⇒ no ^{need.} knowledge design & coding.

⇒ 2 approaches.

1. Equivalence class Partition.

2. Boundary Value Analysis.

EC:

- ① it will check both valide and invalide. the domain of i/p values to the program under test is partitioned into a set of equivalence classes.
- ② No. of Invalide.

Ex.

in V	Va	inv
$-\infty$ to 17	18 to 30	30 - ∞

it divides the statement to equivalence classes.

Ex: 18-30 and 60-80 accepted.

in V	V	inv	V	in V
$-\infty$ - 17	18 - 30	31 - 59	60 - 80	81 - ∞

⇒ NSS - {A, B, C} valid

⇒ ✗ - {A, B, C} - invalide

it is good as testing the code with any other value belonging to the same equivalence class.

2. Boundary Value Analysis:

⇒ it is derived from ECP.

it has 2 boundaries.

Lower.		Upper.	
Below Boundary Value.	above boundary.	Below boundary.	above boundary.
17 invalid	18 valid	30 valid	31 invalid.

⇒ The boundary value analysis is the special case of equivalence class partitioning method that focusses on the boundary of the white box testing: equivalence classes.

⇒ Test both sides of each Boundary (The assumption is that The system behaves differently on either side of boundary).

⇒ A Type of programming error that is frequently committed by Programmers is missing out on the special consideration that should be given to The values at the boundaries of different equivalence classes of inputs.

⇒ The reason behind programmers committing such errors might Purely be due to Psychological factors.

for example, Programmers may improperly use $<$ instead of $<=$ or conversely $<=$ for $<$ etc.

⇒ The equivalence classes that are not a range of Value no boundary value test cases can be Defined.

⇒ The equivalence class that is a range of values the boundary values needed to be Included in the test suite.

సాంఘిక మరియు గణిత సంక్షేమ శాఖల వారి ఉచిత సరఫరా

white-box - Testing :=

⇒ Test internal code & application.

→ it performed by Developers,

⇒ ^{*} class Testing. (or) clean (or) structural (or) transparent.

⇒ 2 approaches.

1. fault based Testing. → 1. Mutation:

2. cover based Testing. 1. Statement.

2. Branch.

3. Multiple Decision.

4. Path.

5. Dataflow.

(1) Fault: (error):

(i) statement cbT:-

execute every statement in a program at least once.

Ex:

```
int cse06(int a, int b) {
```

2.

```
int result = a + b;
```

3.

```
if (result > 0) {
```

4.

```
printf("Positive");
```

5.

```
else if (result < 0);
```

6.

```
printf("negative");
```

7.

```
}
```

test case

Test case: a b

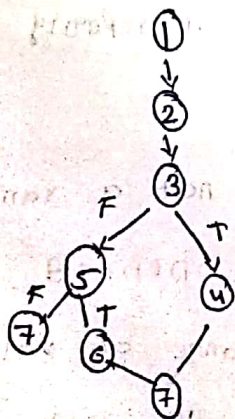
TC 1: 3 4

TC 2: -12 -6

TC 3: 0 0

(2) Branch cbT:-

⇒ CFG (Control flow graph).



case 1: 1 2 3 4 7.

2: 1 2 3 5 6 7

3: 1 2 3 5 7.

2. Multiple Decision:

* This testing is based on analysis of internal structure of the system.

* Glass, clear, transparent, structured.

* It is visible to testers.

* It is applied on all SDLC through.

=> Most of all defects are found in unit, component & integration levels is through white box.

=> Tests are based on coverage of code structure, branches, paths, conditions.

Techniques

stat.

: unit-06 :

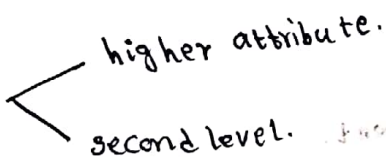
Software Quality:

⇒ exactly what the user wants to do.

⇒ fitness, purpose.

1. Portability: easy to carry (change) ^{from} one place to another.
2. Useability: experts, novice [→] programmers, beginners.
3. Reuseability: Reuse.
4. correctness: removing errors, SRS.
5. maintainability: errors correction, functions adding.

* By using McCall's and ISO 9261 to check the quality of a software.

① McCall's 

② ISO 9261 → Hierarchical.

Software Quality Management system:

2 approaches,

1. Management structure & individual Responsibility.

* whole credit goes to organization.

* organization

↓
QC.

⇒ To control by using below process.

1. Review

2. auditing.

3. Guidelines, procedures.

Evaluation of SLW quality:

⇒ In 5 decades to now there is more change.

⇒ In world war 2 time we got some problems.

1) W.W.2 ⇒ 1939-45.

⇒

it developed by 4 stages.

1. Inspection

↓
2. Quality control

↓
3. Quality assurance

↓
4. TQM

↓
Total quality
Management.

1. Quality control:

To check the problem. To know the effect
of the module.

2. Quality assurance:

To prevent the errors.

SEI CMM:

* software engineering institute capability maturity model.

* it is developed by SEI.

* Carnegie Mellon University → USA → 1987.

* taking help by (US DoD).

↓
Department of Defence.

* helped to organizations to improve the quality of the SW.

* used in 2 ways.

(i) capability evolution :- asses. → to estimate.

(ii) SW process assessment :- improve the quality.

* it is divided into 5 levels.

1. Initial.

2. Repeatable.

3. Define.

4. Managed.

5. Optimizing.

(i) Initial level:-

* ad hoc activities, → suddenly occurring individual persons.

* chaotic :- results.

* no definition (or) process.

⇒ people must there.

(ii) Repeatable

* Management.

* schedule.

* cost.

* FP, COCOMO సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

* Loc.

* early success projects.

* KPA -> management.

Definer

* both process management & development activities.

* user responsibilities.

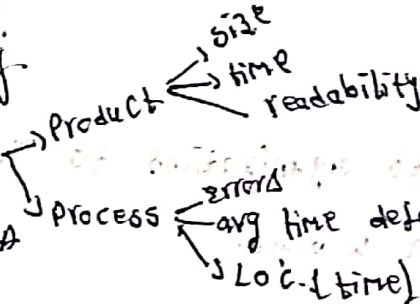
* does not define software quality.

managed: -> process and management. KPA

* Defining software quality

* Metric -> measurement

software quality. KPA



Optimizing:-

-> both product and process.

CPI: continuous process improvement.

=> KPA - Key Process area

Software Reliability:-

* trust and worthness. (or) dependability.

* probability of product working correctly even a given period of time.

* 10% core product.

* 90% non-core product.

* A product contains the large number of errors is called unreliable product.

* Depends on errors. We should know about location of the error.

Reliability high: input data have no errors.

reliability low: if data is no errors occurred.

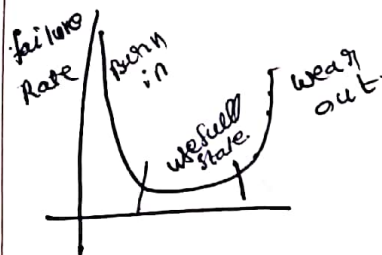
⇒ both user and observer.

⇒ observer dependent.

⇒ it is difficult to find locations. (error location).

⇒ error changes.

Software vs hardware Reliability:-



hardware.



Software Metrics:-

Software Reliability Metrics (Measurements).

⇒ $ROCOF$ ⇒ Rate of occurrence of failure.

⇒ $MTTF$ ⇒ mean time to failure.

⇒ $MTTR$ ⇒ mean time to repair.

⇒ mean time b/w failure ⇒ adding of $mttf$ and $mttr$.

⇒ probability on demand.

⇒ availability. ⇒ uptime.

$$\Rightarrow \text{mean time to failure} = \sum_{i=1}^n \frac{T_{i+1} - T_i}{(n-1)}.$$

$$\sum_{i=1}^n \frac{T_{i+1} - T_i}{n-1}.$$

Short commings :

1. transient
2. permanent
3. recoverable
4. unrecoverable
5. cosmetic

ISO 9000 :-

⇒ 68 countries → 1987

⇒ ISO Series → 3.

1. ISO 9001

2. ISO 9002 → No not design, production

3. ISO 9003 → Testing, Installation.

1. ISO Series 9001:

it gives Designed companies, Production, development, servicing of goods.

it follows 18 sections:

1. management Reliability (4.1)

Quality Policy

2. Quality system (4.2)

Quality standard

3. Contract review (4.3)

4. Design control (4.4)

5. Document control (4.5)

6. Purchasing (4.6)

7. Purchaser supply products (4.7)

8. Product identification (4.8)

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

9. Process control (4.9)

10. inspection & testing (4.10).

11. inspection, measuring, test equipments (4.11)

12. Inspection & Testing status (4.12)

13. Control & non conforming Products (4.13)

14. corrective action (4.14)

15. handling (4.15)

16. Quality Records (4.16)

17. Quality Audits (4.17).

18. Training (4.18)

Guide lines for maintaining the Quality system.

4.1:

⇒ Quality Policy.

⇒ workers should know the Process.

⇒ review & auditing.

⇒ it must have defined document.

4.2:

⇒ should maintain a document for proving the quality.

⇒ A quality system must be maintained & documented.

4.3:

⇒ Reviewing and documenting well defined within time c.

⇒ The organisation has the necessary capability.

4.4:

⇒ Design inputs must be verified as adequate.

⇒ Design must be verified.

⇒ Design changes must be controlled.

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

4.5:

⇒ controlling the document

⇒ documenting in procedures.

⇒ Document changes must be controlled.

4.6:

⇒ Purchasing according to the requirements.

⇒ Purchased material, including bought-in software must be checked for conforming to requirements.

4.7:

Material supplied by a purchaser, for example client-provided software must be properly managed and checked.

4.8:

The product must be identifiable at all stages of the process. In software terms this means configuration management.

4.9:

⇒ The development must be properly managed,

⇒ Quality & requirement must be identified in a quality plan

4.10:

In software terms this requires effective testing i.e.; unit testing, integration testing and system testing

4.11:

If integration, measuring and test equipments are used, they must be properly maintained and calibrate.

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

4.12:

The status of an item must be identified. In software terms this implies configuration management & release control.

4.13:

In software terms, this means keeping untested or faulty software out of the released product.

4.14:

This requirement is both about correcting errors when found and also investigating why the errors occurred.

4.15:

This clause deals with the storage, packing & delivery of the software product.

4.16:

Recording the steps taken to control the quality of the process is essential in order to be able to confirm.

4.17:

Audits of the quality system must be carried out to ensure that it is effective.

4.18:

Training needs must be identified and met.

Silent features of ISO 9001 requirements:

Document control:

All documents concerned with the development of a software product should be properly managed.

సాంఘిక మరియు గిరిజన సంక్షేమ శాఖల వారి ఉచిత సరఫరా

Planning:

Proper plans should be prepared and then progress against these plans should be monitored.

Review:

Important documents across all phases should be independently checked & reviewed for effectiveness & correctness.

Testing:

The product should be tested against specification.

Organisational aspects:

Several organisational aspects should be addressed.

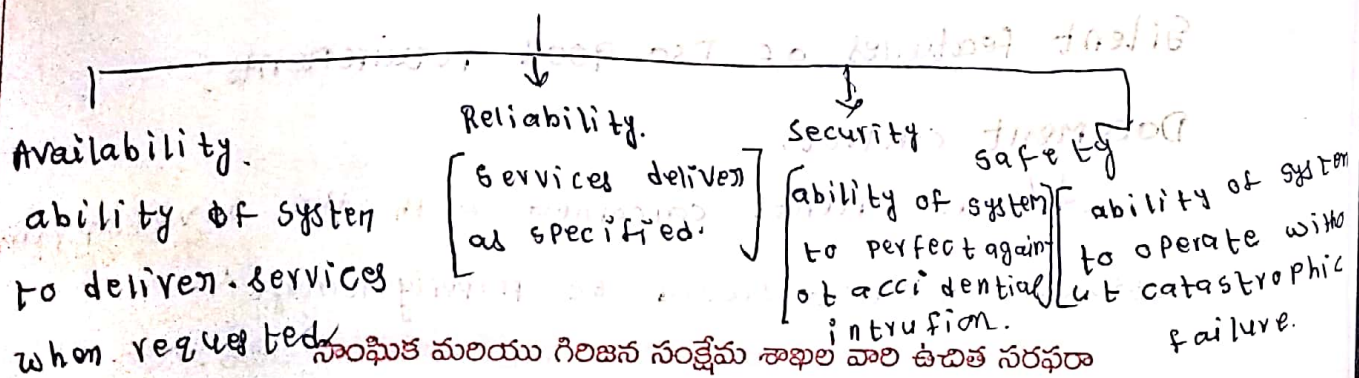
Safe ware risk and safety management:

system safety:

→ systems are part of everyday life so we need to think safety of system.

safety: property of system that reflects system's ability to operate, normally or abnormally without danger of causing human injury and without damage to system's environment.

Dependability:

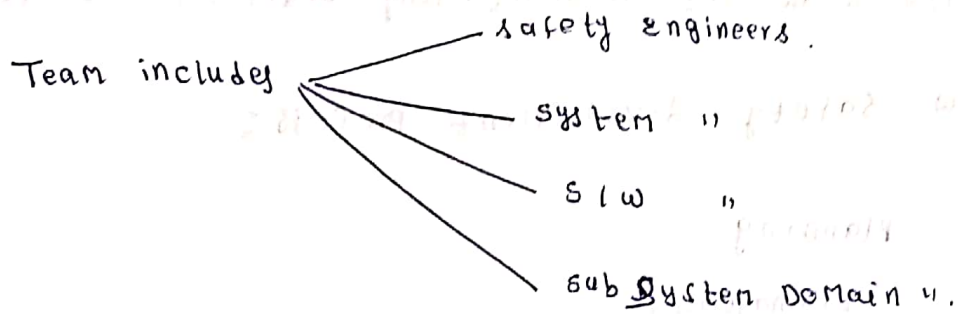


software safety:

* It is system issue so system s/w safety.

* it works when hazards (h/w, s/w or human errors) caused, it have to analysed and solved with good engineering principles.

* It cannot done by isolated engineer so "team" is there



x s/w safety is not only about hazard analyzed report but to influence design of system to ensure safety.

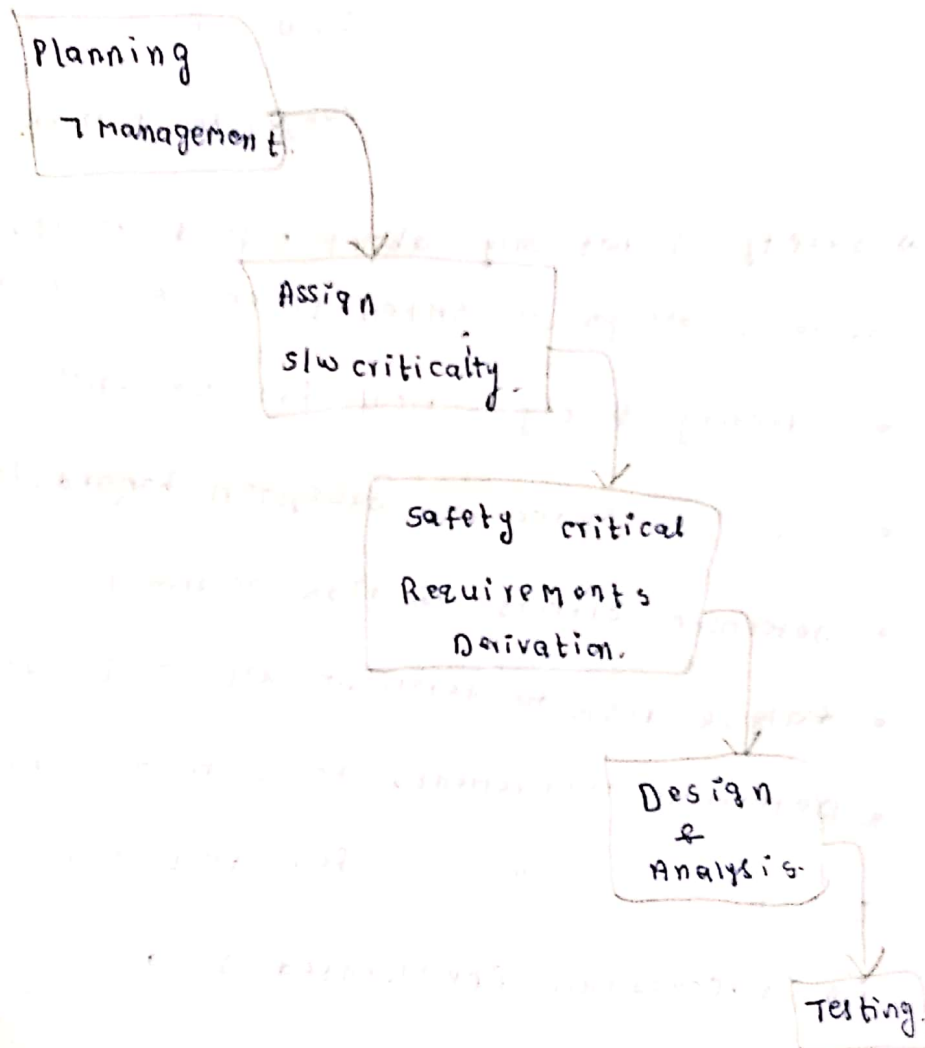
- Identify safety critical fun of system.
- " system or subsystem hazards/risks.
- Determine effects of risk occurrence.
- Analyse risk to determine all contribute faction.
- Determine requirements to eliminate hazard.
- " " for testing to Prove that it is successfull implemented Design.

- communicate with Design team of Program Managements.

Error Failures:

- failure of slw. (not executed / no answer).
- Performing function which are not requiring (wrong ans).
- slw with seq problem (wrong order of execution).

Slw Safety Development Process:



1. Slw Safety Planning & management:

it is first most imp phase before each of life cycle. requirements, Design, coding, testing in cycle.